Credit Card Fraud Prediction Using Naive Bayes and Support Vector Machines

Rahul Padmanabhan
Sr. Fraud Prevention Analyst
FIS
Email: rahul3@mail.usf.edu

***Abstract*—Credit card fraud causes millions of dollars in losses on an annual basis. In this paper I aim to build meta-models for fraud detection using the Naive Bayes approach and the Support Vector Machine (SVM) approach. Two neural network scoring parameters as used as part of our analysis. Real world data from a financial instituion is being tested with separate datasets for training and testing. The analysis on these datasets is based on training data for August 2016. The meta-models are tested on data for September 2016 to analyze model performance in the real world. As fraudulent transactions form a tiny percentage of overall transactions, accuracy rate is not the optimal indicator as the dataset is overbalanced. This paper shows how feature extraction can be used to get better results from a Naive Bayes approach and that SVM does not produce desirable results on overbalanced datasets.**

I. Introduction

Credit card fraud costs the industry million of dollars in losses each year. Often, it is used to fund illegal activities and is considered as a security risk to the industry and the government. While it is a positive aspect that only a small percentage of credit card transactions are fraud, the problem that arises is detecting it and detected the fraud transaction without affecting a lot of valid transactions in the process. Declining valid transactions leads to decreased revenue, customer dissatisfaction and reputation loss for the financial institution. The industry average for fraud is around 15 basis points (bps). In this paper, I am going to build various meta-models to predict fraud on a transaction level using two approaches. One approach is the Naive Bayes method and the other is the SVM method.

A. Background

FIS® (Fidelity Information Services) is the worlds largest credit card processor. FIS® is a Fortune 500 company and is also a member of Standard & Poors 500® Index. For security and privacy purposes, the datasets used contain no account numbers or identifying account specific information. For the same reason, the name of the financial institution whose data this is, is not disclosed. There are three methods of fraud identification from a system perspective:

1) **Switch declines** - at the server when the transaction comes in (e.g. invalid CVV code entered)
2) **Real Time declines** - Custom strategies to decline transactions
3) **Production impact** - Transactions are not declined but attempts to contact the customer are made in order to determine the nature of the transaction (Class)

From a fraud detection standpoint, local models are present and our data includes the scores of two neural networks per transaction. A FICO® generated score and a Visa® generated score. Our dataset takes into account these local parameters and also uses the FICO® score of the prior transaction on the same account (PAD Strat), which is indicated later.

B. Dataset Information

The features I have selected have been carefully chosen to give us the data we need while adequately masking the identity of the customer transacting. Our dataset does not contain customer specific information. The features have been chosen as they are common ones that customized fraud detection strategies are coded on. The description of each field is given in the table below.

Field	Field Description
Trn Typ	Type of Transaction
Dollar Strat	Transaction Amount
Transaction Tag	Valid/Fraud Indicator (CLASS)
Trn Pin Vrfy Cd	Valid/Invalid PIN
Trn Cvv Vrfy Cd	CVV Verification
Trn Pos Ent Cd	Point of Sale Entry
Sic Cd	Merchant Category Code
Mer Cnty Cd	Merchant Country Code
Usr Dat 4	Compromise Association
Cryptogram Valid	EMV indiator for cryptogram validity
Diffzip3	First 3 digit zip code mismatch
Travel_Ent_SIC	Travel and Entertainment Indicator
SIC1	First digit of the SIC code
SIC2	First Two digits of the SIC code
Metro	Metropolitan or rural location indicator
VAA Strat	Visa® Score
Falcon Score Strat	FICO® score
PAD Strat	PAD Score

1) Naive Bayes Datasets: The data set sizes that I used in the Naive Bayes approach are as follows:

Dataset	Valid	Fraud	Total Instances
August (Training)	1,239,752	1,311	1,241,063
September (Test)	1,197,532	1,732	1,199,264

2) SVM Datasets: For the SVM approach, I have used a subset of these datasets. The minority class (Fraud) in the training set (August) has been over-sampled with replacement. The minority class has been over-sampled by 753%. In this approach, I have only considered the transactions that have scored between 800 to 900 using the FICO® neural network score. This was done because SVM is computationally expensive and processing over a million rows in the classifier libsvm in Weka was not possible due to hardware constraints. Another reason for doing so is that we found a larger number of fraudulent transactions within this range. The data set sizes that I have used in the SVM approach are as follows:

Dataset	Valid	Fraud	Total Instances
August (Training)	5,730	1,587	5,916
September (Test)	5,445	230	5,675

C. Evaluation Metrics

As credit card fraud forms only a tiny fraction of the overall transactions, the datasets used in this paper are considered to be imbalanced. An imbalanced dataset is termed as one where the number of positive instances is dwarfed by the number of negative instances. The performance of classifers on imbalanced datasets is diminished because they are designed to generalize from sample data and produce the simplest hypothesis that best fits the data, based on the principle of Occams razor. [1] Due to this, the direct use of the measurement parameter "accuracy" may not be the most prudent approach. An example to justify this would be that if we use a dataset consisting of 1 million transactions and of this, only 1,000 transactions are fraud. If the model chooses to detect all transactions to be valid, the accuracy would be 99.9%. Although 99.9% accuracy may look good in theory, in the real world, the purpose of detecting fraud would be defeated. In this paper, I primarily use accuracy as a metric to check if the valid transactions are being correctly detected. Although I do not get rid of this metric altogther, the value of it is diminished in terms of the intended purpose of this paper.

In all datasets, the field "*Transaction Tag*" is the class field. There are only two classes which are "*Valid*" and "*Fraud*". In our analysis, the definitions of the parameters in the confusion matrix are as follows:

- **True Positive (TP)** - Fraud transactions correctly detected as fraud transactions
- **True Negative (TN)** - Valid transactions correctly detected as valid transactions
- **False Negative (FN)** - Fraud transactions incorrectly detected as validtransactions
- **False Positive (FP)** - Valid Transactions incorrectly detected as fraud transactions

For our analysis, I rely on the following parameters:

Recall

$$recall = \frac{TP}{TP + FN}$$

Precision

$$precision = \frac{TP}{TP + FP}$$

False Positive Rate

$$False\ positive\ rate = \frac{FP}{TP}$$

II. Pre-processing Techniques Used

For the purpose of this paper, I have evaluated the Naive Bayes classifier independently on the entirity of the data avaiable to us. This was done because the Naive Bayes classifer is not as computationally expensive as the SVM(libsvm) classifier. I have then used a subset of our data to compare the Naive Bayes classifier to SVM.

A. Datasets - Naive Bayes Internal Evaluation

For the Naieve Bayes approach, complete August and September datasets have been used. I have used three names to refer to datasets that are used in the internal evaluation of the Naive Bayes classifier. Their names and descriptions are as follows:

- **Original** - Format of data unchanged
- **Custom** - The field Dollar Strat, VAA Strat, Falcon Score Strat and PAD Strat have been grouped into "buckets". An example of this is suppose the transaction amount is in the range of $1 to $25, we categorize it as $1-25. After grouping the four features into these buckets, I have converted the datasets nominal values to binary values in Weka by using NominaltoBinary in the pre-processing section. This process increases the number of features.
- **SMOTE** - This dataset has the same number of instances and data format as the original dataset has. We have not used the "Smoting" technique in this dataset but it is named after it because the idea of using this approach came from reading the paper about "SMOTE" - Synthetic Minority Oversampling Technique [2]. Previous work (Japkowicz, 2000) indicated that over-sampling the minority class could be one of the effective methods of dealing with an imbalanced dataset [3]. In this dataset we have oversampled the minority class (Fraud) with replacement. I used fraud transactions (a total of 12,400) from January 2016 to July 2016 and added them to the dataset by randomly removing 12,400 instances of the valid class.

B. Datasets - SVM

The dataset that I have used in this approach is a subset of the SMOTE dataset above. I only consider the transactions that scored between 800 to 900 using the FICO® neural network score. 10-fold cross validation is performed on the dataset for August and it is tested on the unseen data in September. To implement the *libsvm* classifier we need to ensure that each data instance is represented as a vector of real numbers. We need to convert our nominal data into the numeric format. For example, a four category attribute such as {boxing, basketball, football, chess} would be represented as (0,0,0,1), (0,0,1,0), (0,1,0,0) and (1,0,0,0) respectively. To do this I have pre-processed the data in Weka using the Nominal to Binary approach in the *Preprocess* tab. Per instructions in the documentation, I have proceeded to perform scaling on the data by normalizing the values. Scaling is done in order to reduce the bias that occurs when some of the feature values ranges are of a much higher range compared to those in smaller ranges.

C. Datatsets - Comparison Between Naive Bayes and SVM

Due to the large size of the dataset, I cannot use the datasets that I have used in the Naive Bayes approach for SVM due to the time required for computation. As a result, I have used the dataset that I have used in SVM to evaluate the Naive Bayes classifier for the purpose of comparison i.e. only the transactions that scored between 800 to 900 using the FICO® neural network score in the *SMOTE* dataset. Over sampling by replacement is maintained.

III. Naive Bayes Classifier Approach

The Naive Bayes classifier is a common statistical modeling technique. It allows all the attributes, for the given instance space, to be equally important and contribute towards prediction of the class. The assumption made is that these attributes are independent of each other. Alhough this may not be the case in the real world, this technique tends to outperform more complex classifiers in certain areas of data science. This method is designed for use in supervised induction tasks, in which the performance goal is to accurately predict the class of test instances and in which the training instances include class information [4].The Naive Bayes classifier is based on the Bayes theorem which is:

$$P(H \mid E) = \frac{P(E \mid H)\, P(H)}{P(E)}$$

H is the Hypothesis, E is the given event and P(H|E) represents the probability of H conditional for the given event E.

Kernel Density Estimator - The intuition behind using this method is that it will perform well in domains that violate the normality assumption [4]. Using the kernel density estimate, we estimate the probability density of the classes in a non parameteric manner [5]. In the "Custom" data set for example, I have used the Nominal to Binary method in Weka to pre-process the data. However, there are other features that may not have a normal distribution.

A. Internal Comparison

Before we proceed to the comparison of Naive Bayes vs. SVM, we will analyze the Naive Bayes classifier internally and evaluate its performance on our entire dataset. Naive Bayes classifer is not as computationally expensive as the libsvm classifier so we can evaluate the results on our main dataset with over a million instances. The Naive Bayes classifier has been run in two ways which are:

1) Default
2) With Kernel Density Estimate (KDE)

This is then run on the following datasets:

1) Original Dataset
2) Custom Dataset
3) SMOTE Dataset

This produces six models as a result. 10-fold cross validation is run on these datasets which have the data for August.

TABLE I: August dataset 10-fold Cross Validation

Dataset	Setting	Accuracy	Recall	Precision	FPR
Original	Default	97.41%	0.759	0.03	31.9
Original	KDE	99.57%	0.407	0.106	8.40
Custom	Default	98.00%	0.702	0.036	26.53
Custom	KDE	99.34%	0.461	0.076	12.19
SMOTE	Default	96.75%	0.82	0.229	3.36
SMOTE	KDE	98.66%	0.675	0.434	1.30

Observations on 10-fold cross validation table: Regarding fraud detection, the Naive Bayes - Default classifier on the SMOTE dataset detects 82% of the overall fraud. However, if we also weigh the results of the FPR in our determination of the best performance, we could argue that the Naive Bayes - KDE classifier on the SMOTE dataset is the best performing because although the recall is 0.675 i.e. it detected 67.5% of the overall fraud, it only misclassified 1.3 valid transactions per fraud transaction correctly classified.

TABLE II: September (Test data) Analysis

Dataset	Setting	Accuracy	Recall	Precision	FPR
Original	Default	97.34%	0.737	0.039	24.61
Original	KDE	99.51%	0.359	0.117	7.57
Custom	Default	97.51%	0.725	0.041	23.33
Custom	KDE	99.39%	0.379	0.096	9.37
SMOTE	Default	96.85%	0.761	0.034	28.34
SMOTE	KDE	98.91%	0.556	0.073	12.76

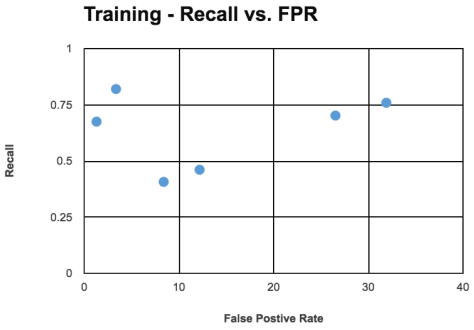

Fig. 1: Training Dataset - Recall vs FPR

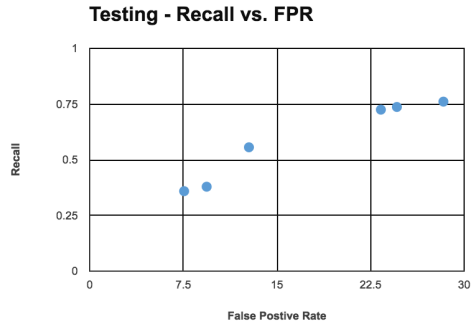

Fig. 2: Test Dataset - Recall vs FPR

Observations: If we were to set a threshold stating that at least 50% of the overall fraud needs to be detected and the FPR below 15 then the Naive Bayes - KDE on the SMOTE dataset would win. It seems to be the most feasible option once we take the valid impact i.e. FPR into consideration. Although the same combination had an FPR of 1.3 on the training set, it has performed far worse on the test set with a decrease in recall and an increase in FPR. In the graphs above, the optimal space is the top left quadrant. We see that while the 10-fold cross validation does well in that regard, the graph on the unseen data does not.

IV. Support Vector Machine Approach

In classification problems, linear models can be very effective and simple. But in real world scenarios, data is seldom linearly separable. Linear classifiers (as the name suggests) fail to classify non-linear boundaries. The limitation of linear models can be addressed by using Support Vector Machines (SVM). SVM is less susceptible to overfitting because of a learned decision boundary called the *maximum margin hyperplane*. It uses a *kernel trick* to do the non-linear mapping. SVM only considers data points close to the hyperplane, treats the instances as *support vectors* and ignores the rest. The data is projected from the original space to a higher dimension space. This allows non-linear boundaries on the data in the original space.

Previous work has identified that support vector machines tend to perform badly on imbalanced datasets [1]. In this paper, for SVM purposes, we use the SMOTE dataset with FICO® scores ranging between 800 to 900 to analyze the overall performance. We have used the libsvm classifier in Weka that was downloaded as separate plugin. In order to gauge which settings are optimal for *libsvm* we have using the *C-Support Vector Classification* (SVM-C) with the following kernels:

1) Linear
2) Radial Basis Function (RBF)
3) Polynomial

A manual coarse grid search has been done to evaluate the model for the optimal values of γ and C. γ is a parameter of the Gaussian Radial Basis function and C is the cost of error. High values of γ are known to produce low variance models and with a high bias and low values of γ produce the opposite. We have done a 10-fold cross validation with the values of gamma and C using a coarse grid search [6]. Due to time constraints, the fine grid search is placed outside the scope of this paper. As this dataset is imbalanced, we assume that high values of C will yield us a low minimum margin separating hyperplane.

Coarse Grid Search on the following:

$\gamma = 2^{-5}, 2^{-4}, 2^{-3}, 2^{3}, 2^{4}, 2^{5}$
$C = 2^{-15}, 2^{-14}, 2^{-13}, 2^{13}, 2^{14}, 2^{15}$

TABLE III: Linear Kernel - 10-fold Cross Validation

Gamma	C	Accuracy	Recall	Precision	FPR
2^{-5}	2^{-15}	73.17%	0	0	0
2^{-4}	2^{-14}	73.71%	0	0	0
2^{-3}	2^{-13}	73.71%	0	0	0
2^{3}	2^{13}	76.35%	0.383	0.591	0.69
2^{4}	2^{14}	75.7%	0.297	0.594	0.68
2^{5}	2^{15}	76.98%	0.271	0.677	0.48

TABLE IV: Linear kernel - Test Data Results

Gamma	C	Accuracy	Recall	Precision	FPR
2^{-5}	2^{-15}	95.94%	0	0	0
2^{-4}	2^{-14}	95.94%	0	0	0
2^{-3}	2^{-13}	95.94%	0	0	0
2^{3}	2^{13}	17.69%	0.943	0.045	0.05
2^{4}	2^{14}	95.11%	0.03	0.115	7.71
2^{5}	2^{15}	94.98%	0.035	0.113	7.88

TABLE V: RBF kernel - 10-fold Cross Validation

Gamma	C	Accuracy	Recall	Precision	FPR
2^{-5}	2^{-15}	73.17%	0	0	0
2^{-4}	2^{-14}	73.17%	0	0	0
2^{-3}	2^{-13}	73.17%	0	0	0
2^{3}	2^{13}	72.07%	0.430	0.477	1.10
2^{4}	2^{14}	71.63%	0.382	0.465	1.15
2^{5}	2^{15}	72.48%	0.334	0.481	1.08

TABLE VI: RBF kernel - Test Data Results

Gamma	C	Accuracy	Recall	Precision	FPR
2^{-5}	2^{-15}	95.94%	0	0	0
2^{-4}	2^{-14}	95.94%	0	0	0
2^{-3}	2^{-13}	95.94%	0	0	0
2^{3}	2^{13}	93.42%	0.026	0.039	0.04
2^{4}	2^{14}	93.85%	0.030	0.053	18.00
2^{5}	2^{15}	94.07%	0.013	0.027	0.03

TABLE VII: Polynomial kernel - 10-fold Cross Validation

Gamma	C	Accuracy	Recall	Precision	FPR
2^{-5}	2^{-15}	73.17%	0	0	0
2^{-4}	2^{-14}	73.17%	0	0	0
2^{-3}	2^{-13}	73.17%	0	0	0
2^{3}	2^{13}	62.12%	0.498	0.354	1.83
2^{4}	2^{14}	60.9%	0.491	0.341	1.93
2^{5}	2^{15}	60.15%	0.471	0.330	2.03

TABLE VIII: Polynomial kernel - Test Data Results

Gamma	C	Accuracy	Recall	Precision	FPR
2^{-5}	2^{-15}	18.73%	0.900	0.043	22.17
2^{-4}	2^{-14}	22.78%	0.887	0.977	21.35
2^{-3}	2^{-13}	30.48%	0.800	0.045	21.19
2^{3}	2^{13}	14.76%	0.952	0.043	22.04
2^{4}	2^{14}	12.26%	0.961	0.043	22.49
2^{5}	2^{15}	11.52%	0.965	0.042	22.58

From this, we observe that the SVM does not have the best performance when it comes to classifying a valid transaction correctly. We notice that when the recall increases, the accuracy decreased beyond an acceptable threshold. SVM performs well on the training set in 10-fold cross validation however, the performance is worse on the test data. Of the three kernels evaluated, the polynomial kernel gives us the best results in terms of recall. However, it runs at a low accuracy rate and high FPR. We also need to keep in mind that using the polynomial kernel is computationally expensive.

V. Comparison of Nave Bayes vs SVM

Previously, in the Naive Bayes analysis, I have considered the entire dataset for August and September which was over a million instances in each case. To accurately compare this approach to SVM we will be using the same dataset that was used for SVM, the SMOTE dataset with FICO® scores from 800-900. We will then analyze the performance of Nave Bayes with and without the kernel density estimator on this dataset. The results are as follows:

TABLE IX: Naive Bayes On The Same Training Data as SVM - 10-Fold Cross Validation

Type	Accuracy	Recall	Precision	FPR
Default	92.59%	0.21	0.118	7.462
KDE	96.29%	0.048	0.176	4.667

TABLE X: Naive Bayes On The Same Test Data as SVM

Type	Accuracy	Recall	Precision	FPR
Default	90.41%	0.117	0.073	12.63
KDE	95.4%	0.004	0.030	32

A. Success Determination

In August, 43% of the overall fraud was declined by existing fraud strategies and the number reduced to 36% in September. This number excludes the *switch declines* that we spoke of earlier. Our focus was on keeping the false positive ratio below 3 and improving the detection percentage to 25% over the existing declined percentage.

VI. Conclusion

In the internal analysis of Naive Bayes, I can conclude that the default (without KDE) has lower accuracy and higher FPR but has higher recall on both the training and testing datasets. The Naive Bayes with KDE has higher accuracy and lower FPR but has a lower recall on both the training and testing datasets.

From the analysis done for the evaluation of Naive Bayes vs. SVM, I can conclude that while both the approaches perform poorly in terms of anomaly detection, the SVM approach performed better than Naive Bayes on the test data set with 10-fold cross validation. The Polynomial kernel with γ of 2^3 and C of 2^{13} detects 49.8% of the overall fraud.

In terms of unseen test data, the SVM approach with RBF Kernel having γ of 2^3 and C of 2^{13} is the most feasible option has it has a recall of 0.026, FPR of 0.04 and accuracy of 93.42%.

From a different standpoint, if we evaluate all the classifiers using only *recall*, the SVM with the polynomial kernel outperforms Naive Bayes as it reaches a maximum recall of 0.965 for the values that we tested i.e. with a γ of 2^5 and C of 2^{15}. The catch to this is that the FPR is 22.58 and accuracy of only 11.52% which in the real world would decline an unacceptable number of valid transactions.

VII. Future Work

For future work, I would like to evaluate other classifiers such as decision trees and evaluate the performance on credit card fraud detection. Due to system hardware limitations, I was not able to evaluate the entire dataset on the *libsvm* classifier. In the future I would like to run a larger sample of the data against the SVM classifier to analyze its performance. In this paper, although I have called the dataset the *SMOTE* dataset, it was done by over sampling of the minority class with replacement. This causes us to lose good examples of the majority class and we fail to utilize those instances for training as a result. I would like to generate synthetic examples without under sampling the majority class in the future. In this paper, I have not evaluated the sigmoid kernel in the *libsvm* classifier. This is another path to explore in the future.

REFERENCES

[1] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," in *European conference on machine learning*. Springer, 2004, pp. 39–50.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[3] N. Japkowicz *et al.*, "Learning from imbalanced data sets: a comparison of various strategies," in *AAAI workshop on learning from imbalanced data sets*, vol. 68. Menlo Park, CA, 2000, pp. 10–15.

[4] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.

[5] A. Pérez, P. Larrañaga, and I. Inza, "Bayesian classifiers based on kernel density estimation: Flexible classifiers," *International Journal of Approximate Reasoning*, vol. 50, no. 2, pp. 341–362, 2009.

[6] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.